DOCUMENT RESUME

ED 388 277 IR 017 426

AUTHOR Muldner, Tomasz
TITLE Rapid Prototyping of Computer-Based Presentations Using NEAT, Version 1.1.
PUB DATE 94
NOTE 8p.; In: Educational Multimedia and Hypermedia, 1994. Proceedings of ED-MEDIA 94--World Conference on Educational Multimedia and Hypermedia (Vancouver, British Columbia, Canada, June 25-30, 1994); see IR 017 359.
PUB TYPE Reports - Descriptive (141) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Authoring Aids (Programming); Computer Literacy; Computer Mediated Communication; *Computer Software Development; *Courseware; *Electronic Publishing; Electronic Text; *Hypermedia; Information Storage; Information Technology
IDENTIFIERS *Links (Indexing); Prototypes

ABSTRACT

NEAT (iNtegrated Environment for Authoring in ToolBook) provides templates and various facilities for the rapid prototyping of computer-based presentations, a capability that is lacking in current authoring systems. NEAT is a specialized authoring system that can be used by authors who have a limited knowledge of computer systems and no programming experience; these authors can communicate with other members of the authoring team using annotations, hypertext links and highlighting. Basic principles of the NEAT design include: maintenance of the book structure, automatic creation of computerized drills using templates for different question types, support for reusability through storage capabilities, a clear display of the final product, user-friendliness, and the ability to inspect scripts and properties of neatware objects. Computer-based presentation developed with NEAT is called neatware. Neatware is a specific type of courseware, based on a book metaphor. Features of neatware include multiple views of the same material, an electronic index, footprints showing student progress, margin and "global" notes, electronic bookmarks, a storage list of previously used pages, hypertext links for non-linear reading, electronic text highlighting, examples, and examples. The user can develop neatware by creating an outline of chapters, sections and pages; various operations can be performed through the Control Panel. The implementation of NEAT is based on a tree data structure in a single text field. (Contains eight references.) (AEF)

**********
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
**********

ED 388 277

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☐ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Gary H. Marks
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

# Rapid Prototyping of Computer-Based Presentations using NEAT, Version 1.1

Tomasz Müldner
*Jodrey School of Computer Science*
*Acadia University, Wolfville, Nova Scotia, B0P 1X0, Canada*
email: tomasz.muldner@acadiau.ca

**Abstract:** NEAT, which stands for iNtegrated Environment for Authoring in ToolBook provides templates and various facilities for the rapid prototyping of computer-based presentations. Outlines of the presentations can be created and modified. NEAT supports reusability by allowing the author to store graphics objects and entire pages in a repository, and to retrieve them when needed. Finally, NEAT provides support for communication between members of the authoring team and between the team and the learners.

## 1 Introduction

Authoring systems have been used for many years and a number of interesting applications (courseware) have been developed, see [MAU90, ALE91]. However, currently existing authoring systems seem to be quite weak in the support for the following areas:

- availability of tools for rapid prototyping
- reusability of concepts and objects, such as graphic objects
- hypertext and hypermedia facilities
- support for building meta-tools, such as icons to execute macros
- extendibility.

While the author is provided with considerable support at the microscopic level, for example specialized editors that are used to create a single frame, text, graphics or animation to appear in this frame, there is little support at a macroscopic level. Thus, it is not possible to prototype courseware and leave details for future development. Another important deficiency of the current authoring systems is that the authoring team has to make an early decision as to what the level of expertise of the expected audience is. For example, there may be beginning learners who do not have any knowledge of the subject, and other learners who have some, or advanced knowledge. The advanced learner would find it boring to go over the entire material necessary for the beginning learner, but may require additional information that is not made available to the beginner. Thus, it would be useful to provide different views of the same material so that each learner can find a view appropriate for his or her background.

Most, if not all, existing authoring systems do not separate the creation and maintenance of the knowledge base, and the presentation of this knowledge (for the description of the second generation of authoring systems, see [MER89]). Members of an authoring team represent different expertise and abilities to work on the development of computer based presentations. Very often there is no single author who creates courseware. Instead, there is an authoring team, whose members represent various types of expertise. For example, one member of the team may be a subject-area expert, with an ultimate knowledge of the subject on which courseware is to be developed. This expert may have limited knowledge or time to actually develop courseware. Other members of the team may be designers, specialists in learning strategies, and programmers;

IR017426



BEST COPY AVAILABLE

for example, programmers specializing in graphics, animations, or multimedia. Only subject-area experts know what should be included in courseware, and so they should prototype this courseware. This prototype can be further refined and modified by other members of the authoring team, provided that there is a simple and effective way of communicating ideas between various members of the team.

NEAT is a specialized authoring system designed to alleviate some of the problems described above. Version 1.0 provided limited support for views and question templates, see [MAY93]. Here, we describe NEAT 1.1, designed and implemented by the author of this paper in the summer of 1993. NEAT can be used by authors who have a very limited knowledge of computer systems and no programming experience. These authors can communicate with other members of the authoring team using annotations, hypertext links and highlighting. Computer-based presentation developed with NEAT is called *neatware*. NEAT is a ToolBook application (ToolBook is produced by Asymetrix), and neatware, produced with NEAT is an application that can be executed with the ToolBook run-time system.

The rest of this paper is organized as follows. First, in Section 2 we describe neatware. Then, in Section 3 we describe NEAT, and in Section 4 prototyping with NEAT. Section 5 briefly describes the implementation of NEAT, and the Conclusion presents several examples of neatware.

## 2 NEATWARE

Neatware is a specific type of courseware, based on a book metaphor. However, neatware can consist of *multiple views*, for example a *beginner* view and an *expert* view. One can think of these views as separate books, and the user can switch between various views. As in a traditional book, each view consists of *chapters*, a *table of contents*, and an *index*. Each chapter consists of sections and pages; a section consists of pages and sections.

As in the real world, the appearance of these books can be altered in several ways by the reader during the learning process. Notes can be kept separate from the book, traditionally in a notebook, known as *global notes*. Notes can be also kept on each page being read, traditionally in the margin of the page, and therefore known as *margin notes*. *Bookmarks* may be placed in various pages of the book. Words on each page can be *highlighted*, in one of several colors.

Unlike in traditional books, an electronic book has other worthwhile features. *Example* pages and *windows* can be accessed from various pages, possibly with a choice of the user's *preference* of example type (e.g. Pascal, C or Modula-2 example), providing more insight into specific details of the neatware. As each page is read, its name is stored in a list known as the *history*, which the reader can use to return to recently visited pages. *Hypertext links* can be used to move around in the book, moving the user to information which may be more relevant. Also, the table of contents contains *bread crumbs*, or *footprints*, which show which pages of the book have already been read. Of course, neatware also includes various *navigation tools* to allow the reader to change pages and access all of the above features.

In summary, each *neatware* has the following characteristics:

- multiple views of the same material; each view consists of chapters; chapters consist of pages
- electronic index which can be modified by the user
- bread crumbs (or footprints), showing the progress of the student
- margin notes that appear on each page
- global notes that resemble a sheet of paper attached to the book
- electronic bookmarks which can be used to save references to the selected pages
- history, showing the list of pages most recently visited by the user
- hypertext links which can be used to read the material in a non-linear fashion
- electronic highlighting of the selected text
- examples which can appear on every page. Each example is associated with the button and can be activated by the learner by clicking on this button. There are three kinds of examples: an example which appears on a separate page; an example which appears in a window on the same page; and an example which can have several appearances, depending on preference that can be selected by the learner (the user's preference may be based on his or her knowledge when studying the book, for example, on knowledge of Pascal or FORTRAN when studying C)
- a repository of examples with a hierarchical structure. These examples can be modified by the user. A sample neatware page is shown in Figure 2.1.

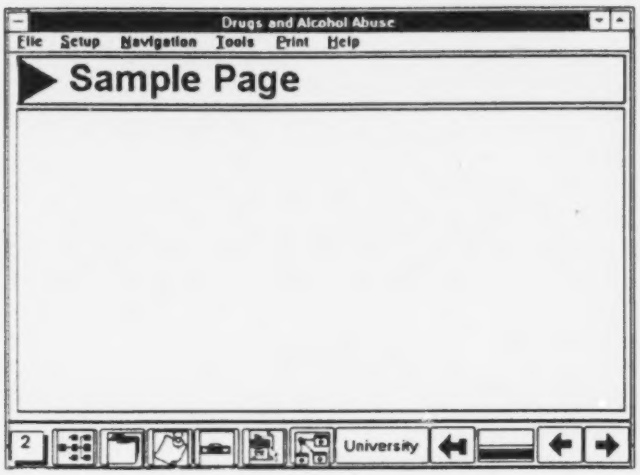



Fig. 2.1 A sample page.

At the bottom of the page, there are various icons, used to navigate through neatware, using one of many navigational features and tools provided by NEAT. In Fig. 2.1 icons represent respectively (from left to right) the page number icon, the table of contents icon, the index icon, the global notes icon, the local notes icon, the highlight icon, the hypertext icon, the view icon, the RETURN icon, the slider icon, and the previous and the next page icons. Menus are provided at the top of the page. Menus are used to perform actions such as "bookmark a page", or "save neatware".

In the rest of this section, we describe two tools used for the communication between the various members of the authoring team and between authors and learners, that is hypertext links and annotations. For more details of neatware, see [MUL93a].

### 2.1 Hypertext Links

Hypertext means a non-linear or associative structure of pieces of text. Pages in neatware may have text which includes the so called *hypertext links*, which are mouse-sensitive text phrases. When such a link is clicked with the mouse, the user is moved to another piece of text, or another page. To help the user find hypertext links, they are underlined. There are three kinds of hypertext links:

- *help* hotword, underlined in blue; produces a window on the same screen
- *forward reference*, underlined in red; move the reader forwards
- *backward reference*, underlined in green; move the reader backwards.

Hypertext links can be created and modified during the development of neatware. Thus, some of them can be used for the communication between members of the authoring team, while others can be left for the learners. Neatware hypertext links can also be created and modified by learners.

### 2.2 Annotations

Annotations are used by the user (both the author and the learner) to highlight important, or difficult sections of courseware. Neatware provides the following types of annotations:

- *highlighting*. A part of the text on the current page can be selected and highlighted by coloring it with one of the available colors. Unlike conventional highlighting, this highlighting can be removed
- *margin notes*, similar to those made on the margin of the page. In version 1.1, margin notes can be stored in both a text and a sound form
- *bookmarks*, which are labels that can be associated with the selected pages. These labels are texts defined at the time the bookmark is created. The user can use existing bookmarks to move to the corresponding pages, add new bookmarks and remove the existing bookmarks. A special bookmark is used when the user quits neatware. This bookmarks stores the current time and date and it can be used to return to the point from which the book was exited
- *bookmarked notes*, which combine bookmarks with margin notes
- *global notes*, similar to those made on separate sheets of paper, and attached to the book.

The next section briefly describes main features of NEAT.



BEST COPY AVAILABLE

## 3 NEAT

The basic principles of the design of NEAT are:

- maintenance of the book structure (that is the structure of chapters, sections, etc.) is transparent to the user. Thus, the user can insert, delete, copy and move chapters, sections, and pages without having to modify any of the navigation tools, such as "go to the next page"
- automatic creation of computerized drills using *templates* for six types of questions; multiple choice, fill in the blanks, numeric analysis, position analysis, matching and textual analysis
- support for *reusability* by allowing the user to store objects and pages for future use
- the development tools do not show up in neatware, so that the user working with NEAT has a clear idea as to what the final product will look like. For this reason, NEAT consists of a series of menus that are used by the user in order to create pages, objects on these pages, etc.
- the user has to switch to the ToolBook's author mode as rarely as possible
- from NEAT, the user can inspect scripts and properties of neatware objects.

There are three tools to support reusability:

- *graphics library* stores graphic objects. The library browser provides two interfaces to the set of objects stored in the library; textual and visual interfaces. The textual interface is in the form of a list of names of all existing objects. The visual interface allows the user to traverse the list of all objects, stored in iconized forms
- *shelves* store pages. For each type of a page, there is a separate shelf. When the user removes a page, it can be stored on a shelf, rather than completely removed. The user can copy or move pages from a shelf into neatware. Shelves are limited to a single book, that is the user can not copy pages from a shelf stored in one neatware into another neatware
- *desks* store pages. Desks are similar to shelves, but they can be used for moving pages between different neatware.

The user can work on at most one neatware at any given time, but it is possible to switch between various neatware. Thus, the user can leave neatware in a certain stage of development, and continue this development at a later time. More details of the NEAT environment are described in [MUL93b]; the next section concentrates on prototyping.

## 4 Prototyping with NEAT

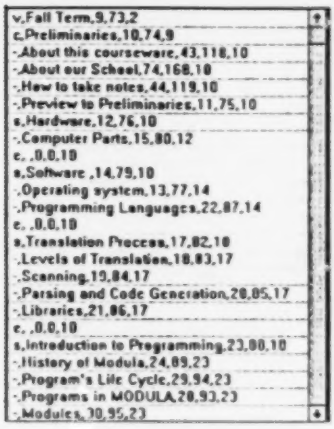
The user can start developing neatware by creating the outline consisting of chapters, sections and pages. Each of these pages may be initially blank or contains only text, and its details can be decided later on. In order to create and maintain the structure of neatware, the user has access to the following operations:

- *go* to the selected page
- *insert* a new page. Here, the user selects the desired type of a page, for example a chapter, a section, or a preview page. The user also selects, in the table of contents, the page *after* which the new page is to be inserted
- *rename* an existing page
- *delete* an existing page. Entire sections, chapters or views can be deleted. The user has an option of storing pages being removed on a shelf, or completely removing them
- *copy* an existing page. Entire sections, or chapters can be copied within the same view, or between different views
- *move* an existing page. Entire sections, or chapters can be moved within the same view, or between different views.

The above operations can be performed through the so-called *Control Panel* page. The author uses this panel to create and modify the structure of neatware. Control Panel is not visible to the learner, who can view the *Table of Contents* page. Below, we describe both types of pages.

### 4.1 Control Panel

The control panel consists of two parts. The field on the left-hand side shows the list of all pages which are currently present in neatware, see Figure 4.1.

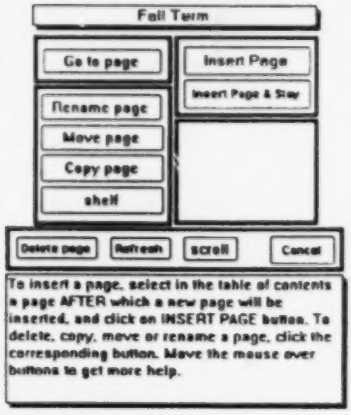



Each line in the above field contains a list. The first character in each line shows the type of the corresponding page:

| | |
|---|---|
| v | view page |
| c | chapter page |
| s | section page |
| - | any other type of a page. |

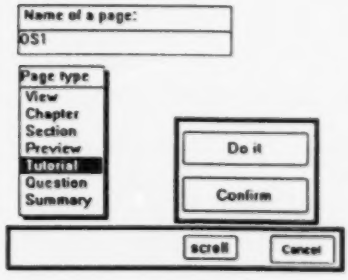
Additionally, the letter "e" indicates the end of the section. The second item on the list on each line is the name of the page. (The remaining items are not relevant for this paper.)
The second half of the control panel, placed on the right-hand side is a set of buttons to perform various operations, see Figure 4.2.

Fig. 4.1 Part of Control Panel


Fall Term
Go to page
Insert Page
Insert Page & Stay
Rename page
Move page
Copy page
shelf
Delete page
Refresh
scroll
Cancel
To insert a page, select in the table of contents a page AFTER which a new page will be inserted, and click on INSERT PAGE button. To delete, copy, move or rename a page, click the corresponding button. Move the mouse over buttons to get more help.

The top button, labeled FALL TERM in the above example, produces a pop-up menu with the names of all existing views. This button can be used to change the current view and show its contents in the field on the left-hand side. For most operations, the user selects the page from the list of all pages. For example, when the user wants to insert a new page, he or she has to specify where the page is to be inserted, by selecting the page *after* which the new page will be included. After selecting the existing page, the user clicks the INSERT PAGE button, and NEAT presents the buttons and fields shown in Figure 4.3.

Fig. 4.2 Another part of Control Panel.




The top field is used to type the name of a new page (OS1 in the above example). The user has also to specify the type of new page by selecting one item in the PAGE TYPE box; by default it is a *tutorial* page. To complete the insertion, the user can either hit RETURN when the text pointer is in the NAME OF A PAGE box, or click on the DO IT button. The user can cancel the operation by clicking the CANCEL button. Clicking the CONFIRM button will allow the user to confirm whether or not the operation is to be performed.

Fig. 4.3 Inserting a new page.



BEST COPY AVAILABLE

When the copy, or move button is pressed, the user has to select the destination view, that is the view into which the selected page, or pages will be copied or moved (even if the destination page is the same as the current page); see for example Figure 4.4. After the user selected the destination view, the left-hand side of the control panel is split into two parts. The upper part of the screen is for the source view

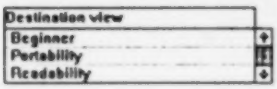



Fig. 4.4 Copying a page.

and the lower part is for the destination view. The user selects the source page or chapter from the upper part of the screen, and then selects a destination page in the lower part of the screen. For example, to copy chapter c1 from view v1 to the beginning of view v2, chapter c1 will be selected on the source screen, and view v2 will be selected on the destination screen.

### 4.2 Table of Contents

A table of contents contains three windows, representing respectively the list of chapters, the list of sections in the selected chapter and the list of pages in the selected section; see Figure 4.5.

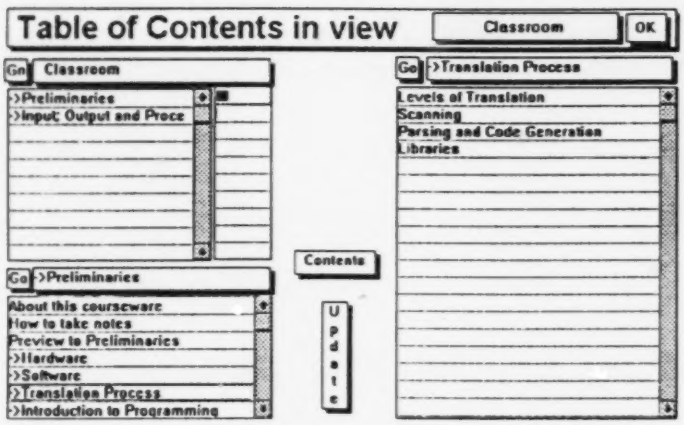



Fig. 4.5 Table of contents.

The user unfolds a chapter or a section by double-clicking it. For each window, there is a title bar, which is also used to unfold the list of pages in this window. For each window, there is also a GO button which is used to move to the selected page. The button placed at the top, right corner labeled OK, moves the user back to the page where the table of contents icon was pressed. The button to the left of this button produces the pop-up menus consisting of the names of all views. The table of contents can be updated by the author after any operation that modifies the structure of neatware; for example after inserting a new page

## 5 Implementation

We provide only a description of the basic data structures to support prototyping. For more details on the implementation of NEAT, in particular the implementation of hypertext, see [MUL93b]; for the description of OpenScript programming language, see [TOO91]. The implementation of NEAT is based on a tree data structure. There are two types of nodes in this tree:

- definition node, which contains a name but does not have any reference to ToolBook pages
- page node, which contains the name and idNumber of a ToolBook page.

The root of the tree is labeled NEAT (and we will refer to this tree as a NEAT tree); each child of the root represents a single view. A subtree rooted at the view's node represents a structure of this view, that is its chapters, sections, etc. Then, to copy a chapter from one view to another view, we perform a (deep) copy of the subtree rooted at the source chapter.

The NEAT tree is implemented in a single text field, using the so-called array implementation of a tree. The line number in the field is referred to as the *node number* for the node stored in this line. A line in the text field allocated for the node N is of the following form:

node number of the father, N (name of the node), list of children



BEST COPY AVAILABLE

where the list of children consists of pairs of the form (name, node number) or (name, idNumber) depending on whether the node represents a definition or a ToolBook page.

The list of available lines in the field is maintained using two properties, representing respectively the maximum number of lines in the field, and the free list, that is the list of lines that have been deallocated. The maximum number of lines in the field is initialized to a certain value, and is extended whenever necessary. This implementation effectively simulates an "open-ended" array. A single most important concept for all navigation operations is that of a node number. Having a node number available, we can find all necessary information such as the page name, its father, or the list of children. Each neatware has a property whose value is the current node number in the NEAT tree. Moving to another page requires not only the execution a ToolBook instruction to go to this page, but also updating the node number. The basic operation used for navigation is a single step in the depth-first traversal of the tree. This is how the user moves to the next page. Moving to the previous page requires a single step in the *reverse* depth-first traversal.

## Conclusion

Our initial experience with NEAT has been very encouraging. During the last tear, the author of this paper was involved in the development of several neatware. The most complete applications are:

- SLADER, neatware on drug and alcohol abuse, see [MUL93c]
- C INTERACTIVE, neatware on teaching programming in C
- MC, neatware on teaching introductory programming in Modula II, used for teaching first year students of Computer Science at Acadia University in 1993/94.

Other neatware are being designed.

During the development of SLADER, we have prototyped large portions of the material, and then presented it to the subject-area experts (specialists on drug and alcohol abuse). We have often used tools such as margin notes, global notes, and hypertext to compile comments provided by these experts and to produce the next version. We have also often modified the structure of neatware; for example the contents of the first version of SLADER has been rearranged as a result of the review conducted by experts from the Counseling Center. Without the support provided by NEAT, we would not be able to complete our work in the same time period.

## Acknowledgments

NEAT was originally designed in the summer of 1992 by Tomasz Müldner from Acadia University, Canada, and Stefan Mayer and Claus Unger from Hagen University, Germany. Version 1.0 of NEAT was implemented by T. Müldner in the fall of 1992, and Version 1.1 was implemented by the same author in the summer of 1993. Several students of Acadia University helped at the various stages of development of NEAT, in particular R. Blondon, G. Poulen, Mark Rhodenizer and B. Santosa. C. van Veen implemented most of SLADER neatware.

## References

[MAU90] Maurer, H., Tomek, I. *Hyper-G - A Survey*. Report 284, IIG, Graz University of Technology, 1990.

[MER89] Merill, D., Li, Z. and Jones M. *Limitations of First Generation Instructional Design*. Educational Technology, Jan. 1990.

[ALE91] Alessi, S. and Trollip S. *Computer-Based Instruction*. Second Edition. Prentice-Hall, 1991.

[MAY93] Mayer, S., Müldner, T. and Unger, C. *NEAT: An iNtegrated Authoring Environment based upon ToolBook*. EDMEDIA'93, Orlando, Florida, June 1993.

[MUL93a] Müldner, T. *NEATWARE Reference Manual*. Technical Report, Acadia University, Sept. 1993.

[MUL93b] Müldner, T. *NEAT Reference Manual*. Technical Report, Acadia University, Sept. 1993.

[MUL93c] Müldner, T., Duncan, P., van Veen C. *Hypermedia Presentation on Drugs and Alcohol Abuse*, Society for Applied Learning Technology, Multimedia'94, Orlando, Florida, Feb. 1994.

[TOO91] *Using ToolBook*. Asymetrix Corporation, 1991.